The **dbexport** and **dbimport** utilities import and export a database and its schema to disk or tape.

The **dbexport** utility unloads an entire database into text files and creates a schema file. You can unload the database and its schema file either to disk or tape. If you prefer, you can unload the schema file to disk and unload the data to tape. You can use the schema file with the **dbimport** utility to re-create the database schema in another IBM Informix environment, and you can edit the schema file to modify the database that **dbimport** creates.

The **dbimport** utility creates a database and loads it with data from text files on tape or disk. The input files consist of a schema file that is used to re-create the database and data files that contain the database data. Normally, you generate the input files with the **dbexport** utility, but you can use any properly formatted input files.

The **dbexport** command unloads a database into text files that you can later import into another database. The command also creates a schema file.

You must have DBA privileges or log in as user **informix** to export a database.

In addition to the data files and the schema file, **dbexport** creates a file of messages named dbexport.out in the current directory. This file contains error messages, warnings, and a display of the SQL data definition statements that it generates. The same material is also written to standard output unless you specify the **-q** option.

During export, the database is locked in exclusive mode. If **dbexport** cannot obtain an exclusive lock, it displays a diagnostic message and exits.

Tip The **dbexport** utility can create files larger than 2 GB. To support such large files, make sure your operating system file-size limits are set sufficiently high. For example, on UNIX, set **ulimit** to unlimited.

Example

The following command exports the table definitions but no data for all the tables in the customer database.

```
dbexport stores_demo -no-data-tables -no-data-tables-accessmethods=customer
```

Example

The following command generates the schema and data for the customer database without the specification of an owner:

```
dbexport stores_demo -nw
```

The **dbexport** utility supports disk and tape destination options.

When you write to disk, **dbexport** creates a subdirectory, *database*.exp, in the directory that the -o option specifies. The **dbexport** utility creates a file with the .unl extension for each table in the database. The schema file is written to the file *database*.sql. The .unl and .sql files are in the *database*.exp directory.

If you do not specify a destination for the data and schema files, the subdirectory *database*.exp is placed in the current working directory.

When you write the data files to tape, you can use the -f option to store the schema file to disk. You are not required to name the schema file *database*.sql. You can give it any name.

UNIX/Linux Only

For database servers on UNIX or Linux, the command is:

```
dbexport //finland/reports
```

The following command exports the database **stores_demo** to tape with a block size of 16 KB and a tape capacity of 24 000 KB. The command also writes the schema file to /tmp/stores_demo.imp.

```
dbexport -t /dev/rmt0 -b 16 -s 24000 -f /tmp/stores_demo.imp
   stores_demo
```

The following command exports the same **stores_demo** database to the directory named /work/exports/stores_demo.exp. The resulting schema file is /work/exports/stores_demo.exp/stores_demo.sql.

```
dbexport -o /home/informix/labs/work/exports stores_demo
```

dbimport:

The **dbimport** command imports previously exported data into another database.

The **dbimport** utility can use files from the following location options:

- All input files are on disk.

- All input files are on tape.
- The schema file is on disk, and the data files are on tape.

The **dbimport** utility supports the following tasks for an imported Informix database server:

- Specify the dbspace where the database will reside
- Create an ANSI-compliant database with unbuffered logging
- Create a database that supports explicit transactions (with buffered or unbuffered logging)
- Create an unlogged database
- Create a database with the NLS case-insensitive property for NCHAR and NVARCHAR strings.
- Process all ALTER TABLE ADD CONSTRAINT and SET CONSTRAINTS statements in the .sql file of the exported database that define enabled or filtering referential constraints so that any foreign-key constraints that are not specified as DISABLED are in ENABLED NOVALIDATE or in FILTERING NOVALIDATE mode.

The user who runs the **dbimport** utility is granted the DBA privilege on the newly created database. The **dbimport** process locks each table as it is being loaded and unlocks the table when the loading is complete.

The input-file location specifies the location of the *database*.exp directory, which contains the files that the **dbimport** utility imports.
If you do not specify an input-file location, **dbimport** searches for data files in the directory *database*.exp under the current directory and for the schema file in *database*.exp/*database*.sql.

Examples showing input file location on UNIX or Linux

To import the **stores_demo** database from a tape with a block size of 16 KB and a capacity of 24 000 KB, issue this command:

```
dbimport -c  -t /dev/rmt0 -b 16 -s 24000 -f
   /tmp/stores_demo.imp stores_demo
```

The schema file is read from /tmp/stores_demo.imp.

To import the **stores_demo** database from the stores_demo.exp directory under the /work/exports directory, issue this command:

```
dbimport -c -i /work/exports stores_demo
```

The schema file is assumed to be /work/exports/stores_demo.exp/stores_demo.sql.

The **dbimport** utility supports options for creating a database, specifying a dbspace for that database, defining logging options, and optionally specifying ANSI/ISO-compliance or NLS case-insensitivity (or both) as properties of the database.

If you created a table or index fragment containing partitions in Informix® Version 10.00 or a later version of the Informix database server, you must use syntax containing the partition name when importing a database that contains multiple partitions within a single dbspace. See the *IBM Informix Guide to SQL: Syntax* for syntax details.

Example showing dbimport create options (UNIX or Linux)

To import the **stores_demo** database from the **/usr/informix/port/stores_demo.exp** directory, issue this command:

```
dbimport -c stores_demo -i /usr/informix/port -l -ansi
```

The new database is ANSI/ISO-compliant.
The next example similarly imports the **stores_demo** database from the **/usr/informix/port/stores_demo.exp** directory. The imported database uses buffered transaction logging and explicit transactions. The **-ci** flag specifies *case insensitivity* in queries and in other operations on columns and character strings of the NCHAR and NVARCHAR data types:

```
dbimport -c stores_demo -i /usr/informix/port -l buffered -ci
```

The **-ansi** and **-ci** options for database properties are not mutually exclusive. You can specify an ANSI/ISO-compliant database that is also NLS case-insensitive, as in the following example of the **dbimport** command:

```
dbimport -c stores_demo -i /usr/informix/port -l -ansi -ci
```